

# CSS PART 2

## Module 4

Ryan Collins

# **ABSOLUTE POSITIONING**



# ABSOLUTE POSITIONING

## NORMAL FLOW

Default positioning

Element is in flow

Elements are laid out according to their order in HTML code

Default positioning  
`position: relative`

## ABSOLUTE POSITIONING

Element is removed from the normal flow: “out of flow”

No impact on surrounding elements, may overlap other elements

We can use top, bottom, left, or right to offset the element from it's relatively positioned container

`position: absolute`


```
position: static;
```

```
position: relative;  
top: 40px; left: 40px;
```

```
position: absolute;  
top: 40px; left: 40px;
```

```
position: sticky;  
top: 20px;
```

In this demo you can control the `position` property for the yellow box.



To see the effect of `sticky` positioning, select the `position: sticky` option and scroll this container.

The element will scroll along with its container, until it is at the top of the container.

# PSEUDO-ELEMENTS

# PSEUDO-ELEMENTS

```
<title>Pseudo</title>
<style>
  body {
    font-family: sans-serif;
  }
  h1 {
    text-transform: uppercase;
    font-size: 22px;
  }
  h1::first-letter {
    font-size: 52px;
    color: #4f46e5;
  }
</style>
</head>
<body>
  <div>
    <h1>Very Long Title</h1>
    <p>
      Lorem ipsum dolor sit amet, consectetur adipisicing elit. Natus
      recusandae accusantium possimus error velit dolorum voluptas
      officiis,
      optio repellendus quis suscipit tempora doloribus. Dicta
      recusandae
      nobis ut assumenda blanditiis. Fugiat!
    </p>
  </div>
</body>
</html>
```

Can you guess what will happen in this example?

<https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements>

# **3 WAYS OF LAYOUTS**



# WHAT DOES LAYOUT MEAN?

## LAYOUT

Layout is the way text, images and other content is placed or arranged on a webpage

Layout gives the page a visual structure, organizing our content in a defined visual representation

Building a layout: We arrange page elements into a visual structure, instead of having them placed one after another in the normal document flow

### IKEA Food



#### IKEA Food Click and collect

Click and collect is now available for all of your favourite IKEA Food products found in our Swedish Food Market. Like our classic Swedish meatballs, sustainably sourced salmon, refreshing beverages, and delicious desserts.

[Shop IKEA Food Click and collect](#)



#### IKEA Family Food offer

IKEA Family members can purchase two packages of PÅTÅR organic coffee (250g) for \$9.99. Regular price \$5.99 each.

Price valid July 1 to August 31, 2022. Offer available in-store only.

[Learn more about IKEA Family](#)



#### IKEA Family Food offer

IKEA Family members can purchase two packages of SJÖRAPPORT Salmon fillet (500g) for \$24.99. Regular price \$14.99 each.

Price valid July 1 to 31, 2022. Offer available in-store only.

[Learn more about IKEA Food](#)

### Explore the beautiful possibilities






# PAGE LAYOUT VS. COMPONENT LAYOUT

PAGE  
LAYOUT


### IKEA Food



**IKEA Food Click and collect**

Click and collect is now available for all of your favourite IKEA Food products found in our Swedish Food Market. Like our classic Swedish meatballs, sustainably sourced salmon, refreshing beverages, and delicious desserts.

Shop IKEA Food Click and collect




**IKEA Family Food offer**

IKEA Family members can purchase two packages of PÅTÅR organic coffee (250g) for \$9.99. Regular price \$5.99 each.

Price valid July 1 to August 31, 2022. Offer available in-store only.

Learn more about IKEA Family



**IKEA Family Food offer**




IKEA Family members can purchase two packages of SJÖRAPPOR Salmon fillet (500g) for \$24.99. Regular price \$14.99 each.

Price valid July 1 to 31, 2022. Offer available in-store only.

Learn more about IKEA Food

### Explore the beautiful possibilities

Shop all products



## Pricing

Quickly build an effective pricing table for your potential customers with this Bootstrap example. It's built with default Bootstrap components and utilities with little customization.

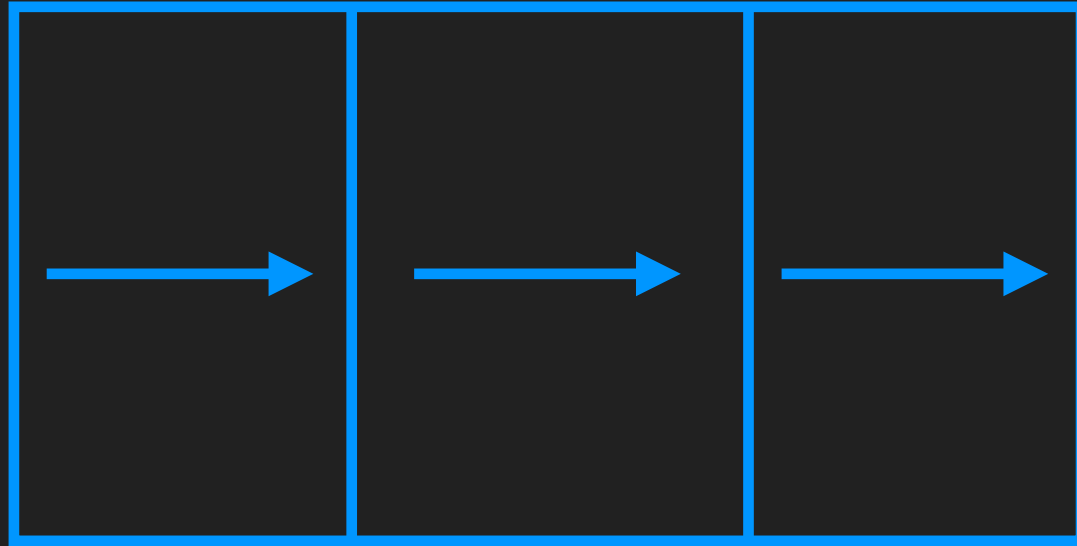
Free	Pro	Enterprise
<h3>\$0 / mo</h3> <p>10 users included 2 GB of storage Email support Help center access</p> <p>Sign up for free</p>	<h3>\$15 / mo</h3> <p>20 users included 10 GB of storage Priority email support Help center access</p> <p>Get started</p>	<h3>\$29 / mo</h3> <p>30 users included 15 GB of storage Phone and email support Help center access</p> <p>Contact us</p>

COMPONENT LAYOUT



# 3 WAYS OF BUILDING LAYOUTS WITH CSS

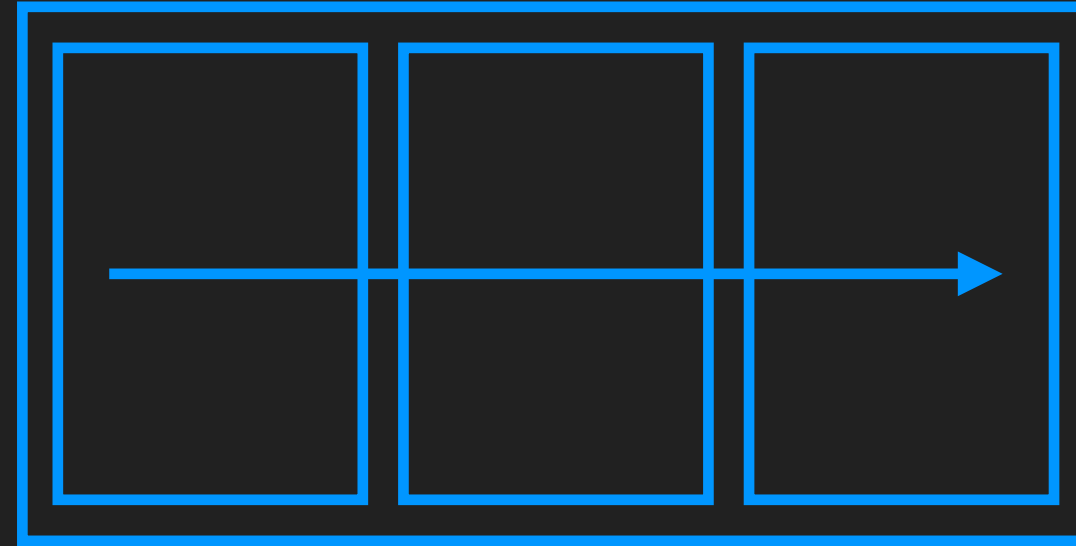
1



FLOAT LAYOUTS

The old way of building layouts of all sizes, using the float CSS property

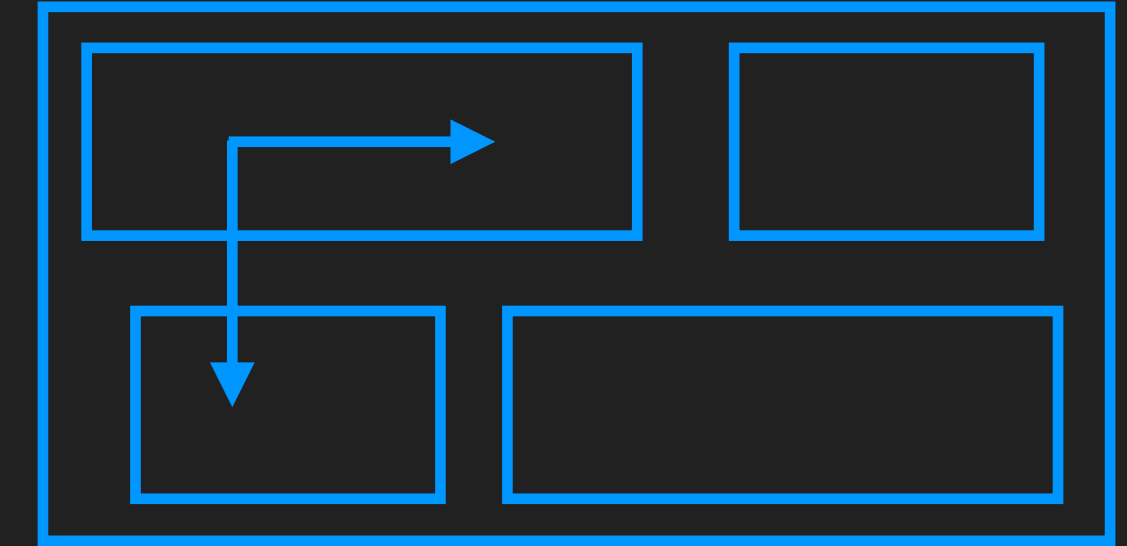
2



FLEXBOX

Modern way of laying out elements in a 1-dimensional row without using floats. Perfect for component layouts

3



CSS GRID

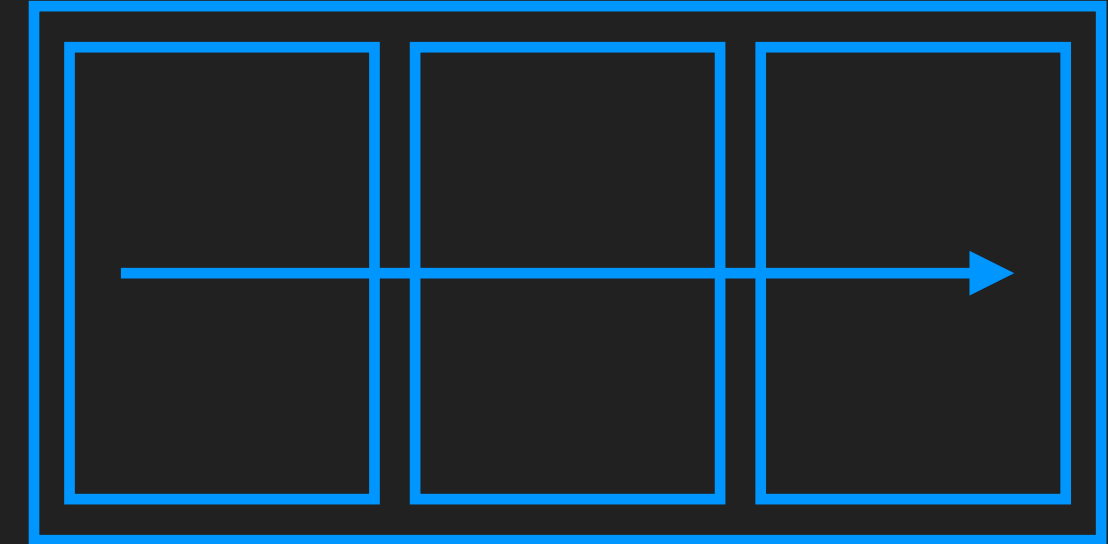
For laying out an element in a fully-fledged 2-dimensional grid. Perfect for page layouts and complex components

# **FLEXBOX**



# WHAT IS FLEXBOX?

## FLEXBOX



Flexbox is a set of related CSS properties for building 1-dimensional layouts

The main idea behind flexbox is that empty space inside a container element can be automatically divided by its child elements

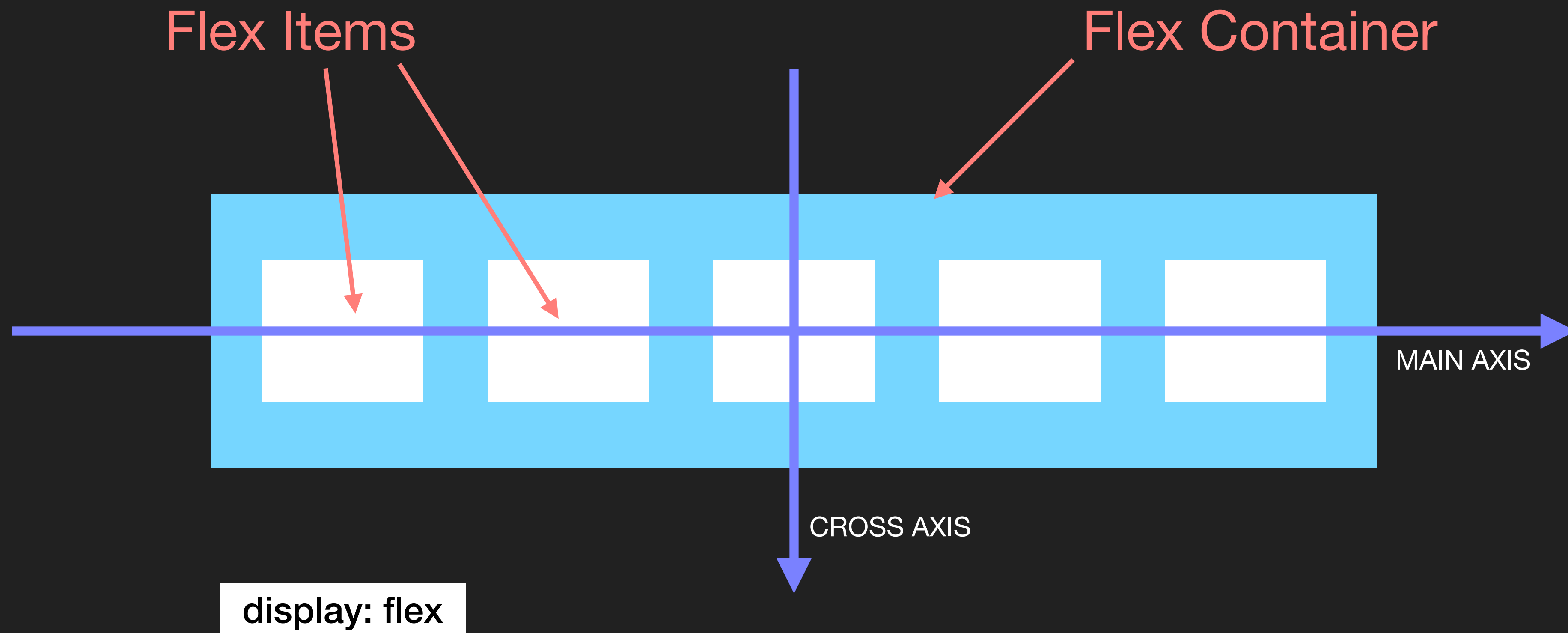
Flexbox makes it easy to automatically align items to one another inside a parent container, both horizontally and vertically

Flexbox solves many common problems, such as vertical entering and creating equal-height columns

Review MDN documentation first

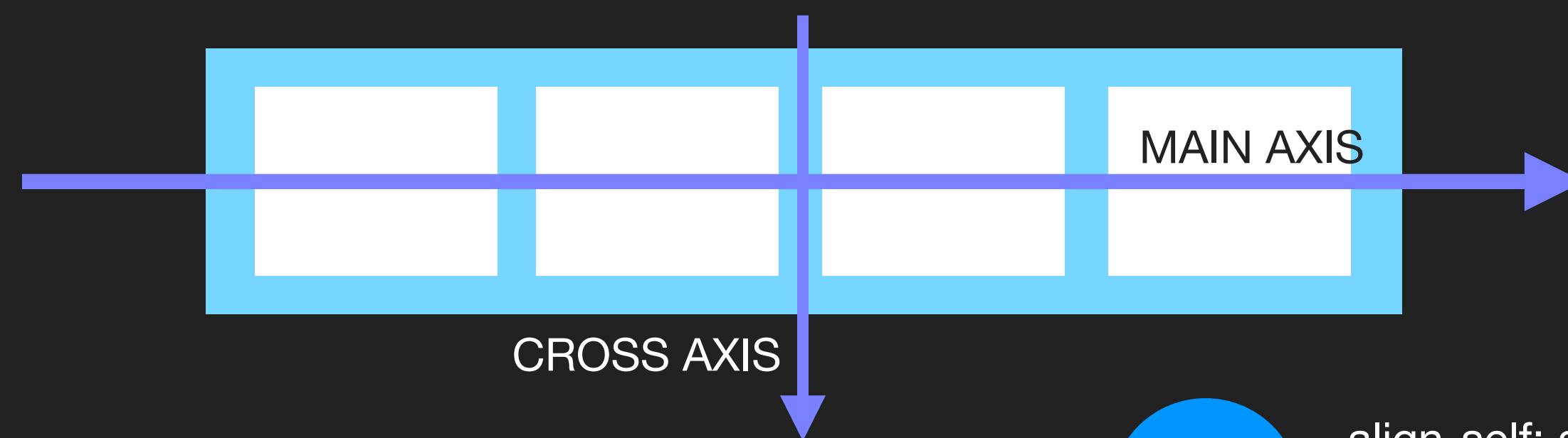
[https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS\\_layout/Flexbox](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox)

# FLEXBOX TERMINOLOGY





# FLEXBOX



# FLEXBOX

1

gap: 0 | <length>

To create space between items, without using a margin

2

justify-content: flex-start | flex-end | center | space-between | space-around space-evenly

To align items along the main axis (horizontally by default)

3

align-items: stretch | flex-start | flex-end | center | baseline

To align items along the cross axis (vertically by default)

4

flex-direction row | row-reverse | column | column\_reverse

To define which is the main axis

5

flex-wrap: nowrap | wrap | wrap-reverse

To allow items to wrap into a new line if they are too big

6

Align-content: stretch | flex-start | flex-end | center | space-between | space-around

Only applies when there are multiple lines

1

align-self: auto | stretch | flex-start | flex-end | center | baseline

To overwrite align-items for individual flex items

2

flex-grow: 0 | <integer>

To allow an element to grow ( 0 means no, 1+ means yes )

3

flex-shrink: 1 | <integer>

To allow an element to shrink ( 0 means no, 1+ means yes )

4

flex-basis: auto | <length>

To define an items width, instead of the width property

5

flex: 0 1 auto | <int> <int> <len>

Recommended for shorthand flex-grow, -shrink, -basis

# INTRO TO FLEXBOX

HTML

and

CSS

are

amazing

languages

to

learn

Practice in flexbox source code!



# SPACING AND ALIGNING

HTML

and

CSS

are

amazing

languages

to

learn

Practice in flexbox source code!

# THE FLEX PROPERTY

REVIEW

[https://developer.mozilla.org/en-US/docs/Web/CSS/Shorthand\\_properties](https://developer.mozilla.org/en-US/docs/Web/CSS/Shorthand_properties)



# INTERACTIVE ACTIVITY

## BUILDING A FLEXBOX LAYOUT

TRY THE FOLLOWING EXAMPLES

[https://www.w3schools.com/css/css3\\_flexbox.asp](https://www.w3schools.com/css/css3_flexbox.asp)

[https://www.w3schools.com/css/css3\\_flexbox\\_container.asp](https://www.w3schools.com/css/css3_flexbox_container.asp)

[https://www.w3schools.com/css/css3\\_flexbox\\_items.asp](https://www.w3schools.com/css/css3_flexbox_items.asp)



# INTRO TO GRID

Review docs

[https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Grid\\_Layout](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout)

# GRID OVERVIEW



# WHAT IS CSS GRID?

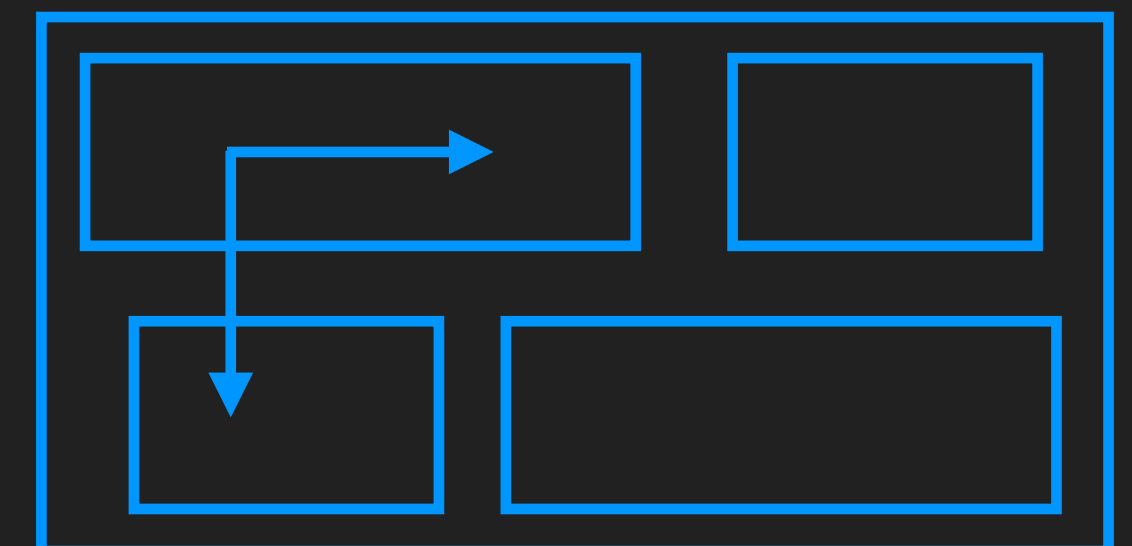
## CSS GRID

CSS grid is a set of properties for building 2-dimensional layouts

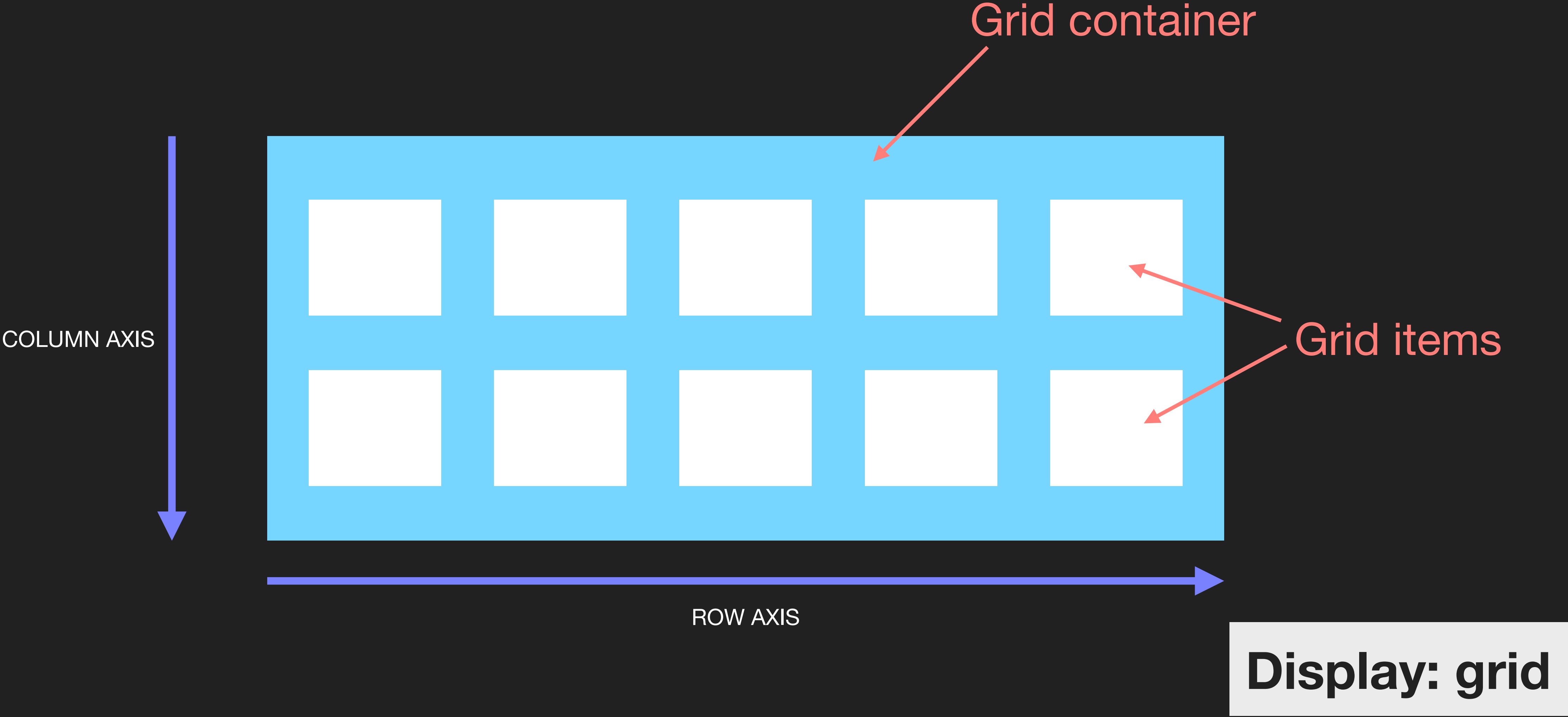
The main idea behind CSS Grid is that we divide a container element into rows and columns that can be filled with child elements

In two-dimensional contexts, CSS Grid allows us to write less nested HTML and easier to read CSS

CSS Grid is not meant to replace flexbox, instead they work perfectly well together. If you need a 2D layout you can use CSS Grid



# BASIC GRID TERMINOLOGY



# MORE GRID TERMINOLOGY

Grid track

Grid lines

COLUMN AXIS

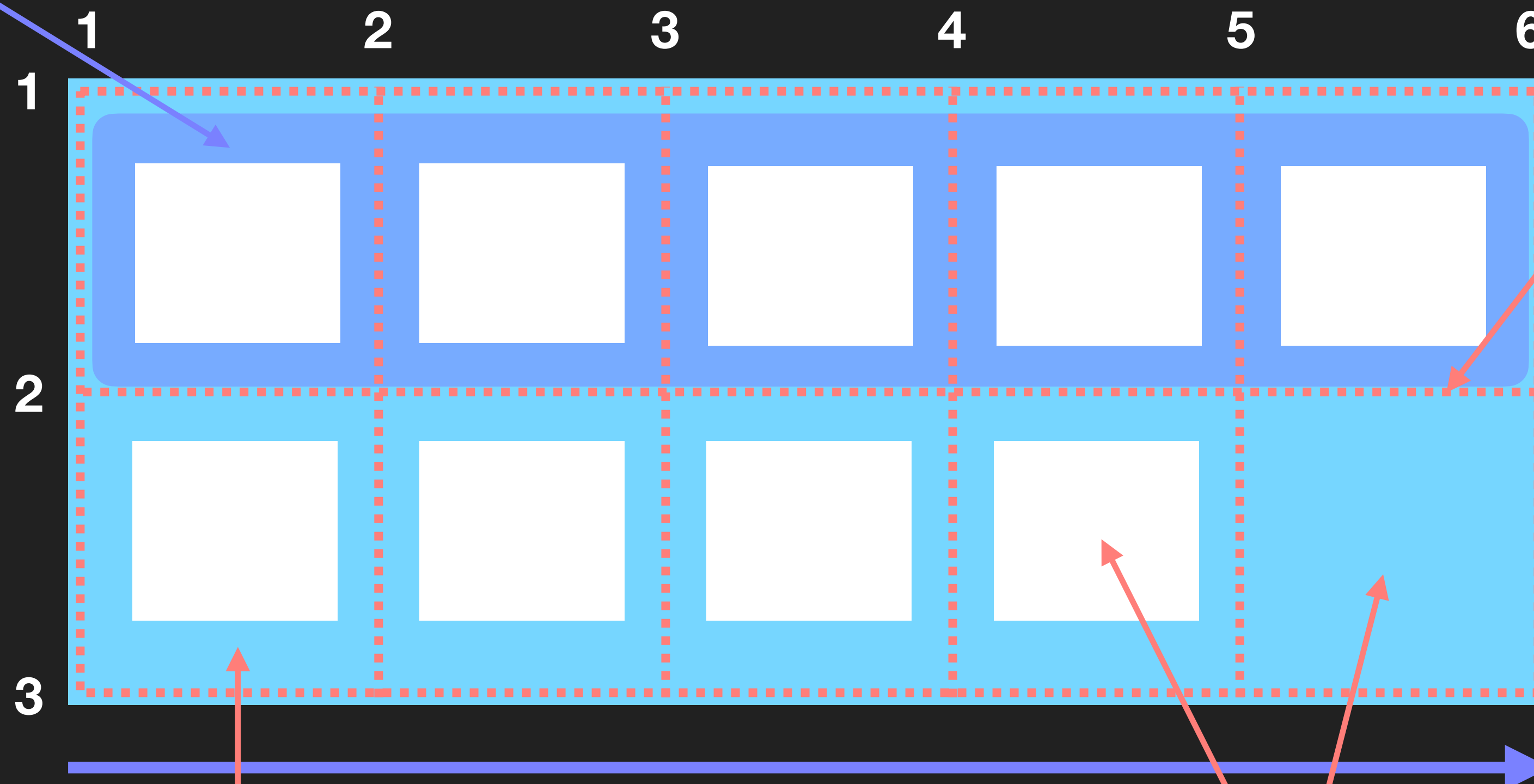
ROW AXIS

Grid gap

Grid cell

I know this is a lot to take in, so  
don't worry about this, if it  
doesn't make sense right now

Grid cells can be filled by a  
grid item, or be empty






# GRID CONTAINER

1

grid-template-rows: <track size> \*  
grid-template-columns: <track size> \*

To establish the grid row and column tracks. One length unit for each track. Any unit can be used, new, new unit fills unused space

2

row-gap: 0 | <length>  
column-gap: 0 | <length>  gap: 0 | <length>

To create empty space between tracks

3

justify-items: stretch | start | center | end  
align-items: stretch | start | center | end

To align items inside rows / columns (horizontally / vertically)

4

justify-content: start | center | end  
align-content: start | center | end

To align entire grid inside the grid container. Only applies if the container is larger than the grid.

<https://developer.mozilla.org/en-US/docs/Web/CSS/grid>

# GRID ITEMS

1

grid-column: <start line> / <end line> | span <number>  
Grid-row: <start line> / <end line> | span <number>

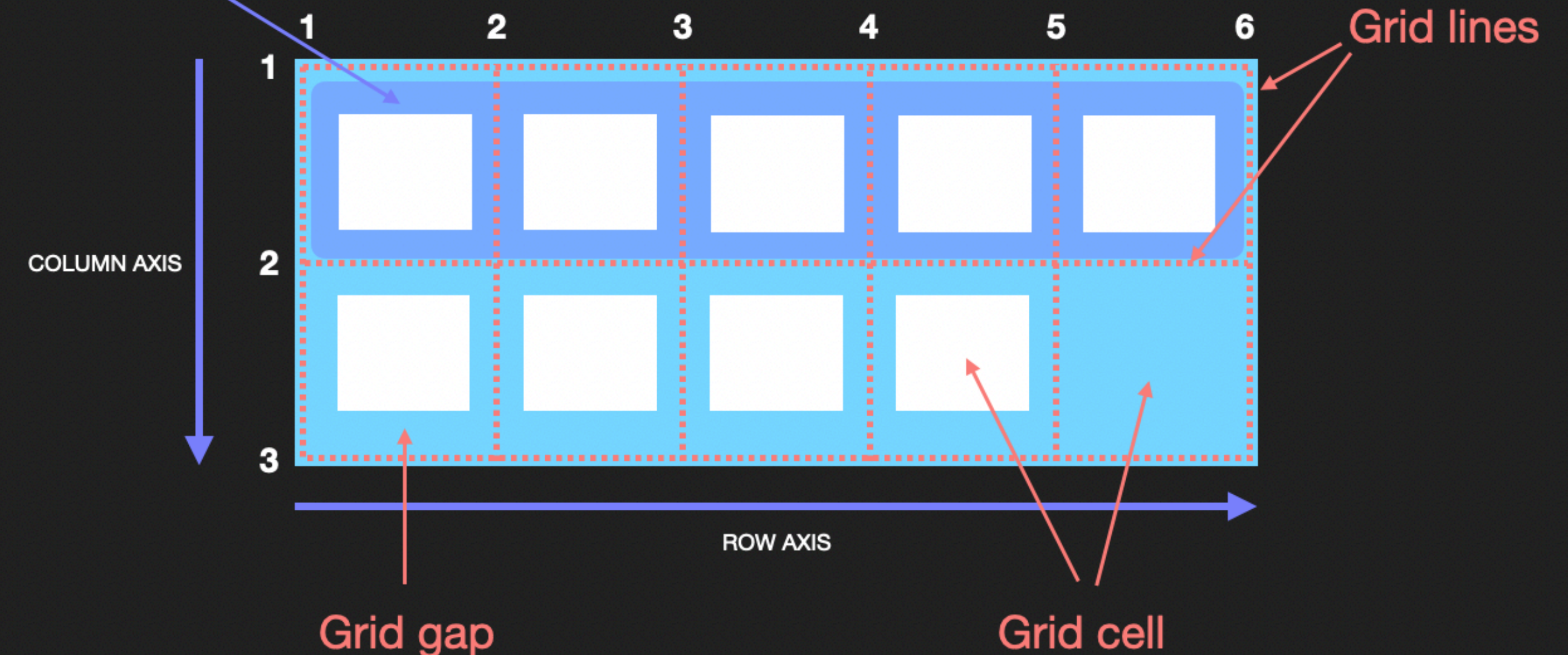
To place a grid item into a specific cell, based on the line numbers. Span keyword can be used to span an item across more cells

2

justify-self: stretch | start | center | end  
align-self: stretch | start | center | end

To overwrite justify-items / align-items for single items

Grid track



# INTRODUCTION TO CSS GRID

(1) HTML

(2) and

(3) CSS

(4) are

(5) amazing

(6) languages

(7) to

(8) learn

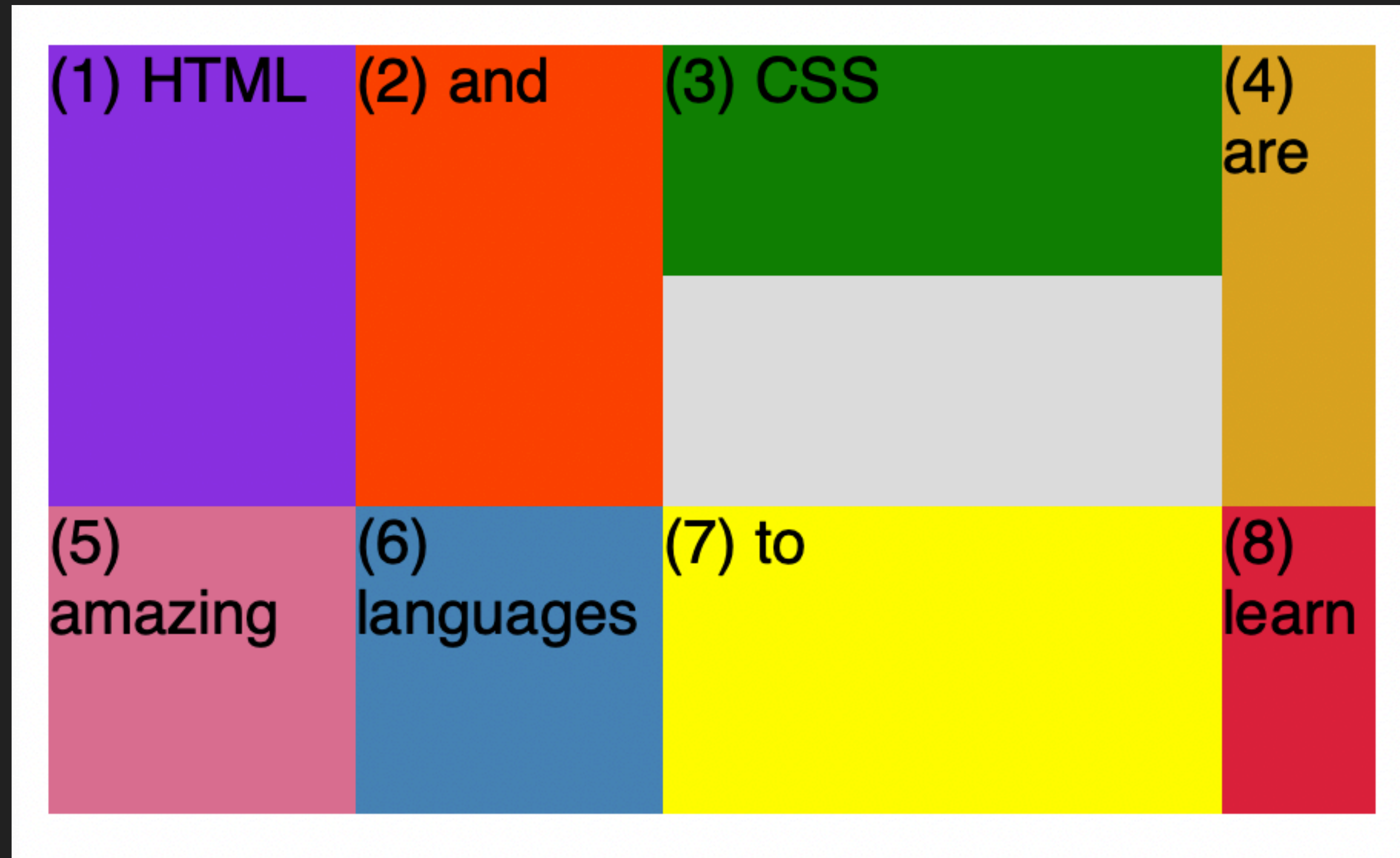
Practice example

Check source code name `css-grid.html`

# GRID COLUMNS AND ROWS



# GRID COLUMN AND ROW SIZING



See sizing-grid-.html example

<https://developer.mozilla.org/en-US/docs/Web/CSS/grid-template-columns>

# PLACING AND SPANNING GRID ITEMS

# **ALIGNING GRID ITEMS AND TRACKS**



# BUILDING A GRID LAYOUT